A Simple Host Deflection Scheme for High-Speed LANs Using Wormhole Routing[†]

Po-Chi Hu and Leonard Kleinrock
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095-1596

Abstract

Wormhole routing is a simple, low-cost switching scheme often used for supercomputer interconnections. Recently, it also has been applied to high-speed local area networks to support applications demanding high-data-rate communications, such as cluster computing. The drawback to wormhole routing is its low link efficiency caused by worm blocking. To overcome this blocking problem, a timeout scheme was investigated in [3] by analytical modeling. In this paper, we present timeout simulation results, showing the effect of packet size, propagation delay, and network size. Furthermore, a simple deflection scheme, which we call host deflection, is introduced and tested. This simple host deflection scheme requires only small modifications to the protocol and very little processing power from the switches; it improves the network throughput significantly.

1. Introduction

Wormhole routing is a common switching scheme for supercomputer intercommunications. It has the merits of low latency, low cost, and easy implementation. In addition to the supercomputer interconnections, these merits also make wormhole routing attractive to high-speed local area networks (LANs). One example is Myricom's Myrinet [11], which has been adopted as the LAN infrastructure for the Supercomputer SuperNet (SSN) project conducted by researchers at UCLA, JPL and Aerospace Corp [5, 6, 7].

A local area network (LAN) using wormhole routing technology has several advantages over conventional LANs (such as FDDI, DQDB and Ethernet), or an *Asynchronous Transfer Mode* (ATM) LAN. It can provide higher data rate than those others, using a simple switch structure. It also exhibits very low network latency, because wormhole routing employs cut-through [4], which minimizes the transmission delay (as shown in figure 1) without the necessity of packetization; the overhead of *segmentation-and-reassembly* (SAR) is saved as well. Wormhole routing works on an arbitrary network. The aggregate network bandwidth grows as more switches are added. In wormhole routing, circuit setup is not required; by using *source routing*, switches are not required to handle routing or circuit setup; packets (which are also called *worms*) are routed according to the specified routing path, which is determined by the source host and attached to the packet header. As a consequence, a wormhole routing switch is inexpensive, compared to an ATM switch, while still providing high bandwidth[1].

However, a high-speed LAN requires not only low latency but also very high throughput, which is not easy to achieve with wormhole routing because of the blocking problem. Blocking occurs when there are two packets contending for the same link; one of them has to be stalled, which consequently reduces the efficiency of links that have been occupied by the blocked packet. This degrades the achievable network throughput. To overcome this throughput limitation, several good ideas, such as *adaptive routing* and *virtual channels* [1, 2, 8, 10] have been proposed and studied for the supercomputer interconnection environment. Unfortunately, adaptive routing is not suitable for wormhole routing LANs, because it is not easy to implement for irregular network topologies of LANs. Consequently, adaptive routing needs intelligent switches to reconstruct the routing path, a requirement that

[†]This work was supported by the Advanced Research Projects Agency, ARPA/CSTO, under Contract DABT63-93-C-0055 "The Distributed Supercomputer Supernet — A Multi Service Optical Intelligent Network".

[1]For example, an 8×8 *Myrinet* switch that can support 640Mbps on each port is estimated to cost only $2,400.

0-8186-7453-9/96 $05.00 © 1996 IEEE

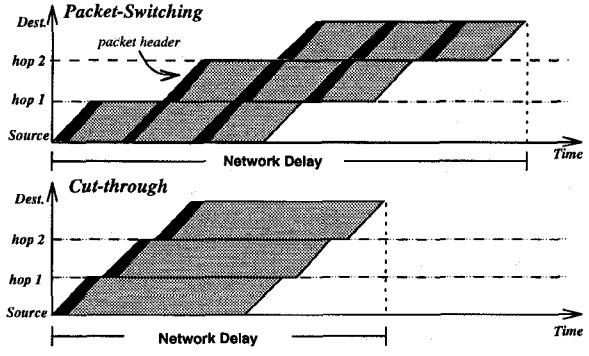

Figure 1. The delay comparison of store-and-forward packet switching and cut-through.

will increase the cost of switches significantly. On the other hand, virtual channels improve network throughput modestly but increase network delay due to multiplexing among virtual channels on the same physical link.

To search for a better approach to the throughput problem of wormhole routing LANs, we studied a timeout scheme through an analytical model in our preceding paper [3], and showed that it is effective when the network size is limited. In this paper, we first review wormhole routing and the simulator we developed, in section 2. Then, in section 3, we present simulation results of the timeout study, which reveals the performance characteristics of the timeout scheme with respect to the packet size, propagation delay, and network size (in terms of the number of switches and hosts). These results indicate that the timeout scheme, which performs well when the network size is small, cannot gain much in network throughput as the network size grows due to the exponentially decreasing probability of successful transmissions. Therefore, in section 4, a simple deflection scheme, which we call *host deflection*, is proposed and investigated by simulation. This simple host deflection scheme requires little protocol modification and processing power from switches, but improves the network throughput substantially. Finally, section 5 contains the conclusion and future work.

2. Wormhole routing

In general, we consider a network for which all communication links are bi-directional with the same capacity. Packets are generated and absorbed at hosts only. We measure packet length by *flits*, which is the amount of data that can be transmitted in one clock cycle (defined to be the unit of time). For example, the 640Mbps Myrinet has one byte per flit lasting 12.5ns.

Wormhole routing was first introduced in [12]. It was developed from the earlier idea of *cut-through switching* [4]. In wormhole routing, switches have relatively small buffers. As opposed to store-and-forward switching, as soon as a packet header (or its routing information) is received, this packet is forwarded to the next switch before it is completely received (see figure 1); if the outgoing link to the next switch is busy serving another packet, the packet gets blocked and resides in the switch until the outgoing link is available. In this case, called *blocking*, the switch must inform the previous up-stream switch to stop transmission (i.e., it exercises *back-pressure flow control*) due to the limited size of buffers, as shown in figure 2. A packet might be buffered in several nodes along the chain while stuck in the middle of the network due to blocking. With wormhole routing, deadlocks are possible unless a deadlock-free routing strategy is employed. A survey of wormhole routing can be found in [9].

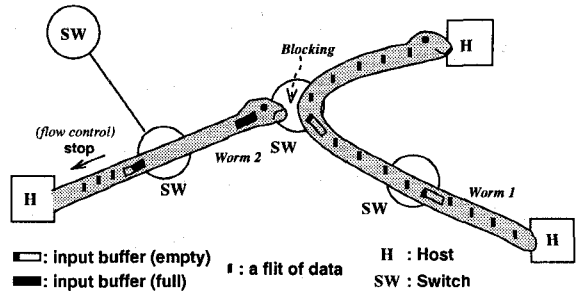

Figure 2. An illustration of wormhole routing.

Back-pressure flow control depends upon tracking the space left in the input buffers, which is associated with each input port of the switches (figure 2). These buffers are primarily used to accommodate data currently in transit, due to the non-zero propagation delay in LANs. Buffers operate with two important parameters: the *low-threshold* and the *high-threshold*. Whenever the available buffer space is less than the low-threshold, it sends a **STOP** signal to inform the up-stream node to stop transmitting. On the other hand, whenever the available buffer space is higher than the high-threshold, a **GO** signal will be sent up-stream to resume the transmission. The gap between the low-threshold and the high-threshold reduces the

number of flow control signals.

Source routing is employed since switches have little processing power and cannot determine the routing path for worms. A routing path, which specifies the links that a worm will traverse in order, is generated by the source host and attached to the head of the packet. Since switches have no intelligence (for low cost) and specifically can do no adaptive routing, routing paths do not change except at hosts when timeout retransmissions occur[2].

Backward timeout reset is the basic mechanism we use to solve deadlock and throughput problems. Whenever a worm head reaches a switch, a timer starts counting how long this worm resides at this switch while waiting for its outgoing link to become available (thus advancing to the next switch or host node). If this "residence time" exceeds a timeout threshold, then a timeout event is triggered; a switch at which timeout occurs will then clear all buffers occupied by this worm and will issue a timeout reset signal backward to the upstream node from which this worm came. A switch which receives a timeout reset signal will pass this signal further upstream and will also free the outgoing link and any buffer occupied by this timed-out worm. This process continues until the timeout reset signal reaches the source host where the worm was generated. (We assume that a switch can always send the timeout reset signal upstream even if the tail of the worm has already left this switch). The source host, after receiving the timeout reset signal, will stop the transmission of this worm if the transmission is still in progress, and will insert the worm back into the tail of this host's packet queue so that it will be retransmitted later. A timeout example is illustrated in figure 3.

The Simulator performs *discrete-event* simulations at the flit level with the following assumptions:

- Worms are generated as a Poisson process, and their size has an exponential distribution.
- Worm generation rates are identical at all hosts. Moreover, the distance to a host is uniformly chosen from among all feasible distances. Hosts at the same distance from the source are selected as the destination according to a uniform distribution.
- All possible shortest paths are equally chosen by the routing procedure.
- Bandwidth consumed by flow control and timeout signals is negligible.

[2]A retransmitted worm that is timed-out in previous try may attempt a different path to prevent a repeated timeout at the same place.

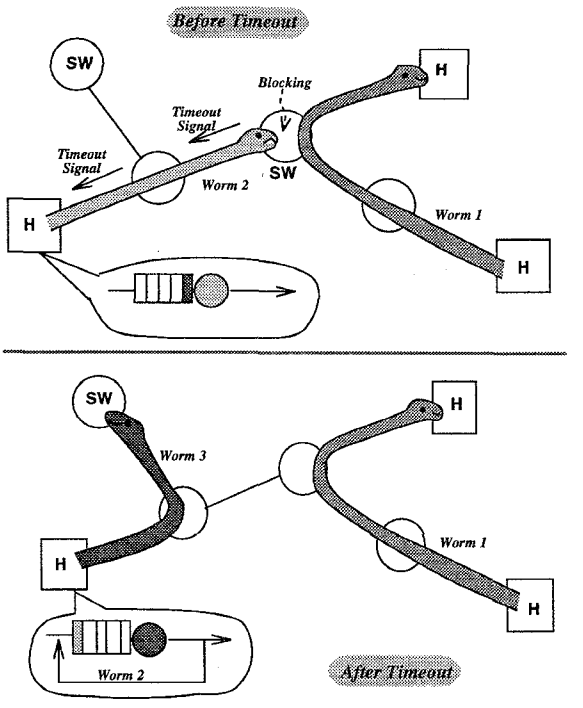

Figure 3. An example of timeout. The upper figure shows a network snapshot before the timed-out worm is rejected. The lower one is after the rejection.

- *First-come-first-serve* (FCFS) discipline for resolving link contentions.
- To avoid implementation dependent details, we assume that the size of a packet header is not changed during transmission[3].

Some of the simulation results were verified by the analytical model in [3].

To make the study less complicated, we use a *torus* as the network topology due to its nice symmetry property. We assume that each switch has eight in/out ports and four of them are connected to hosts. A 3×3 example is shown in figure 4, in which there are 9 switches and 36 hosts. In all simulations, the link propagation delay is set to 10 units of time, which corresponds to a link length of 22.5 meters in Myrinet[4]. Finally, the low-threshold and high-threshold of input buffers are set as 27 and 43 respectively. The buffer

[3]With source routing, one piece of the routing information will be stripped off at each hop. Consequently, the packet header becomes shorter as it advances in the network.

size is 80 flits.

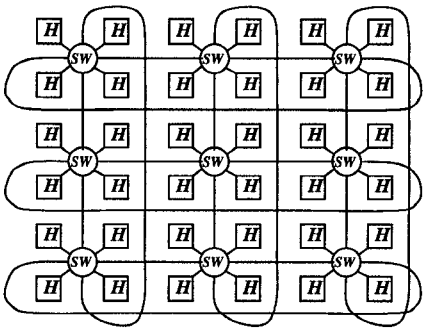

Figure 4. An example network configuration: 3×3 torus.

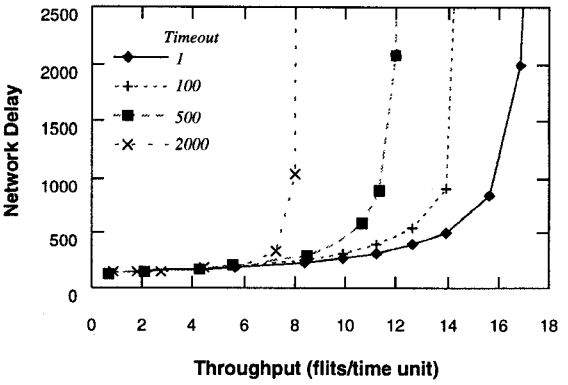

Figure 5. Throughput vs. network delay with different timeout values (3×3 torus, propagation delay = 10 time units, average worm size = 100 flits).

3. Results of timeout simulations

Timeout improves network performance by stopping the ineffective waiting of a blocked worm. As illustrated in figure 3, rejecting those blocked worms makes the occupied links available; these links may then serve other worms that can possibly reache their destinations without difficulty. Timeout also alleviates *head-of-line* blocking at the host. Persistent waiting (blocking) of the original wormhole routing procedure does not allow this flexibility.

Using simulations with different timeout values, we show the power of timeouts. Figure 5 indicates that a short timeout value can increase the maximum network throughput to a factor of two, compared to the long timeout case. It also shows no increase in delay for the use of timeouts. To find the optimal timeout value with respect to different network parameters (such as, worm size, propagation delay, and network size), simulations were run with very high traffic loads for various parameter sets. Figure 6 shows that the optimal timeout value increases as the packet size decreases. Also, a large worm size results in high throughput. In contrast, a short link propagation delay lessens the optimal timeout value and increase the maximum throughput, as shown in figure 7.

The above phenomena can be understood through figure 8. As shown in this figure, when the packet size is large, it is more likely that blocking will last long; therefore, a smaller timeout is better. Similarly, a long propagation delay causes the timeout to waste more bandwidth; hence waiting longer is preferred. Figure 9 points out why a larger packet size results in higher maximum network throughput; it is simply because a long worm has relatively less overhead in attempting to reach its destination.

However, the simulation results show that the maximum throughput does not grow proportionally to network size (figure 10). The aggregate network throughput might even be worse in the large network case. This can be explained by the probability of a worm successfully reaching its destination. Obviously, this probability decreases exponentially as its path becomes longer. That is,

$$P_S = (1 - P_T)^h$$

where P_S is the probability of a successful transmission, P_T is the probability of timeout at a switch node[5], and h is the path length, (measured in number of hops).

From the probability of a successful transmission, we can easily find N, the average number of transmissions required for a worm to reach its destination, as:

$$N = \sum_{i=1}^{\infty} iP_S(1 - P_S)^{i-1} = \frac{1}{P_S} = (1 - P_T)^{-h}$$

We find that N increases exponentially with the path length, h. As a result, this enormous number of retransmissions eliminates any gain due to timeouts, and makes it ineffective for large networks. To overcome this effect in large networks, we propose a simple *host deflection* scheme in the next section.

[4]The *myrinet* link bandwidth is 640Mbps. Each flit is one byte of data, which gives us the time unit, 12.5ns. However, the propagation velocity in a Myrinet cable is about 0.6c, where c is the speed of light. Hence, $0.6c \times 12.5ns = 22.5m$.

[5]For simplicity, we here assume that the timeout probability is identical at all switches.

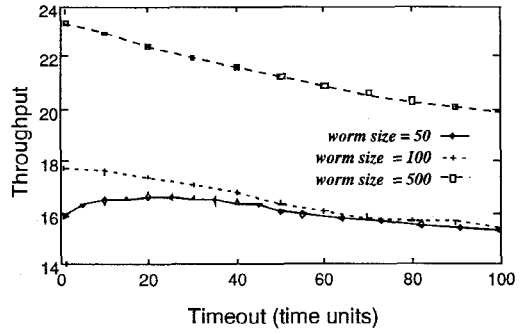

Figure 6. Throughput vs. timeout with different average worm sizes (3×3 torus, propagation delay = 10 time units).

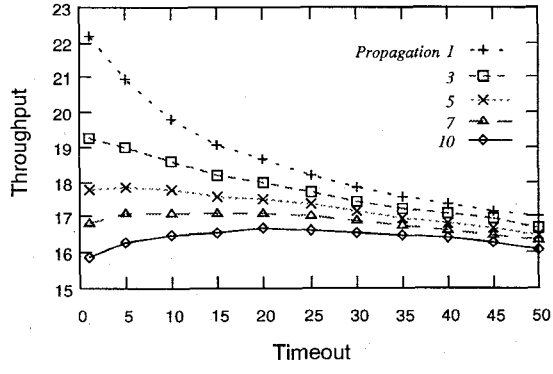

Figure 7. Throughput vs. Timeout with different link propagation delays (3 × 3 torus, average worm size = 50 flits).

4. The host deflection scheme

The basic idea of the host deflection scheme is to use connected hosts as temporary buffers to perform store-and-forward as illustrated in figure 11. We assume that the amount of memory in a host (i.e., a workstation, or a server...) is large and will rarely be filled by deflected worms. The host deflection scheme works as follows:

1. When a worm arrives at a switch, the switch starts a counter which measures the amount of time that this worm sojourns at this switch.

2. The worm keeps probing its outgoing link until the link is free or until the counter exceeds the timeout value.

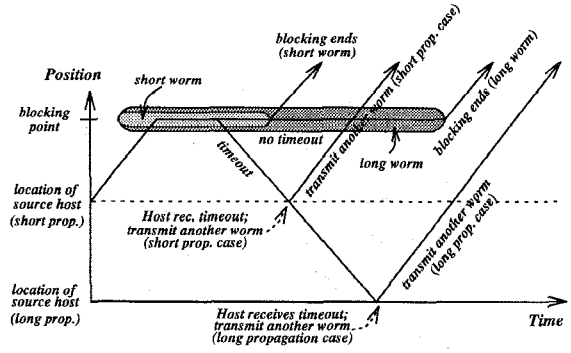

Figure 8. The worm size and propagation delay change the performance of timeout. The arrows show the dynamics of a blocked worm.

3. If the worm finds its outgoing link free, it immediately seizes the link and advances to the next node. Also, the switch disables the counter so that timeout will not occur.

4. When the counter exceeds the timeout value, in which case timeout occurs, the worm probes all links connected to a local host, and chooses one of them that is not currently being used as the deflection path. If there is no available path, the switch discards this worm and sends a timeout signal backward to the up-stream node.

Therefore, instead of discarding a blocked worm, the switch may simply deflect it to a connected host which will temporary buffer the worm in its local memory and retransmit it later along the same link back to the switch. By using deflection, two goals are accomplished. First, we save the retransmissions caused by timeout at the worm's source host (which could be a long distance from the timeout node). Second, using a host as a store-and-forward buffer, we release links that would have been frozen by the blocked worm and make them available for others; hence we avoid the waste of bandwidth due to the transmission break of blocked worms.

However, it should be noted that the deflection scheme increases the traffic load on the switch-to-host links (the cost of deflection). If the packet size is large, this deflection cost may be more than the timeout cost, which is caused by the retransmissions. If we use C_T to denote the cost of a retransmission, we have,

$$C_T = \alpha \times \mathcal{T} \times h$$

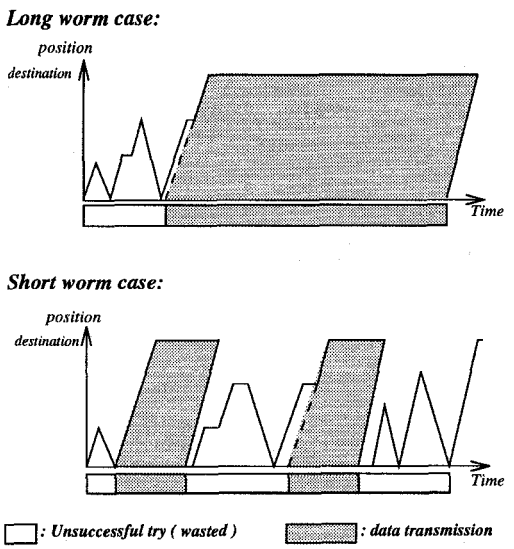

Figure 9. The process of transmissions. In both the long and short worm cases, the expected time for a worm head to reach the destination is similar. Since the payload is larger for longer worms, the efficiency is higher.

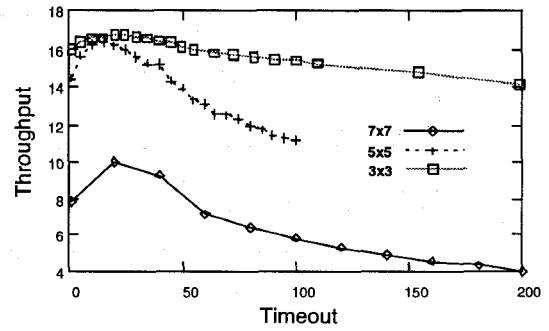

Figure 10. Throughput vs. timeout with different network sizes. (average worm size = 50 flits, propagation delay = 10)

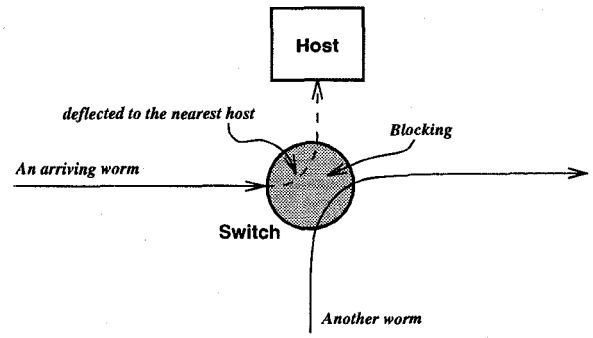

Figure 11. An illustration of the host deflection scheme.

where $\mathcal{T}$ is the amount of time spent on each hop (which is proportional to the timeout value), and h is the distance between the switch and the source host, in terms of numbers of hops. α is simply a multiplicative factor. Because the cost for a retransmission is proportional to the distance between the switch and the source (not to the worm size, unless the whole worm is on the path), no deflection is preferred when the timed-out worm is close to its source host, or is quite large. One simple example of this case is when a worm is timed-out at the first hop. In this case, the cost of "timeout-then-discard" is the retransmission from one hop away (from the source host), while the cost of "timeout-then-deflect" is the retransmission from one hop away (from the buffering host) plus the extra load on the switch-to-host link. Clearly, the latter cannot beat the former, unless hosts are unequally loaded. When hosts are unequally loaded, the deflection scheme may possibly balance the load distribution and consequently improve the network performance.

To show the effect of the switch-to-source distance, a parameter which is called *"i-hop prohibited"* is introduced. We say the deflection is "i-hop prohibited" if host deflection only occurs when a worm has advanced more than i hops from its source host. This parameter is used to prevent excess host deflection when a worm is too close to its source or its size is too large.

In figures 12 and 13, the maximum network throughput is plotted versus the timeout values. This leads to the following conclusions:

- The host deflection scheme improves the network throughput significantly. It is about four times better in throughput (43:10, for the case, of an average worm size = 50 flits). Considering the efficiency of a switch-to-switch link (which is the bottleneck for the network throughput), it is clear that the degradation of network performance due to the increase of network size is completely resolved by the host deflection scheme, as shown in table 1.
- As we expected, the longer the worm, the higher the hop-prohibited for the best network perfor-

	3x3 no deflection	3x3 with deflection	7x7 no deflection	7x7 with deflection	7x7 with input buffering[6]
aggregate throughput	16.7	16.57	12	44	32.6
link efficiency	0.463	0.46	0.18	0.67	0.5

Table 1. A comparison of the aggregate throughput and link efficiency (average worm size = 50 flits).

mance.

- The higher the timeout value, the lower the hop-prohibited for the best network performance. This is because longer timeouts increase the cost of a retransmission.
- Almost all the best performance occurs with a very short timeout value.

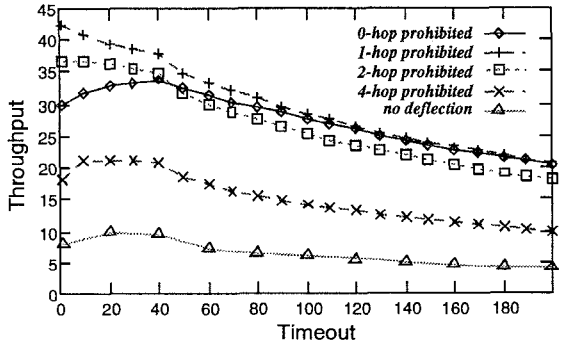

Figure 12. Throughput vs. timeout. (7×7 torus, average worm size = 50 flits)

Actually, with host deflection, the whole routing path is broken into several segments interleaved with store-and-forward nodes (the buffering hosts). It resembles crossing several small wormhole routing networks where the timeout scheme works best. The comparisons in table 1 also indicate that networks with host deflection outperform cut-through network with infinite size input buffers at the switches, which is quite impressive.

The last conclusion suggests a new idea on how to modify the deflection scheme. Because the best performance usually appears when the timeout value is very

[6] With infinite size of input buffers, no timeout, no deflection.

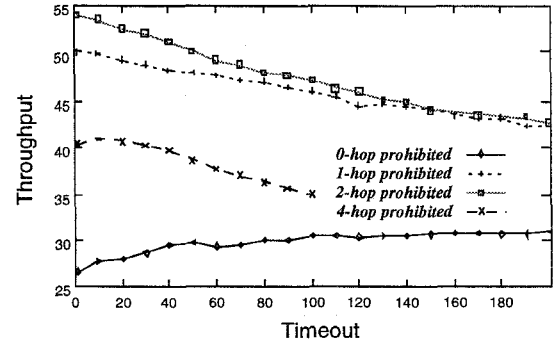

Figure 13. Throughput v.s. timeout. (7×7 torus, average worm size = 1000 flits)

short, it may be better to deflect a blocked worm as soon as there is a deflection path available. Hence, we let a blocked worm keep trying to find a deflection path until it is timed-out. The simulation results shown in figure 14 indicate a moderate throughput improvement with this modification.

The complexity for doing host deflection is low. First, switches need only know which links are connecting to a host; this could be done either manually, or by sending a control signal to the switch from hosts at the link hook up time or periodically. Second, source routing is preserved without any increased complexity. The deflection host simply retransmits the deflected worm back into the network, and the rest of its route need not change at all. No computation for re-routing is required, but the improvement of the network performance is dramatic as shown in figures 12 and 13.

5. Conclusion and future work

In this paper, we discussed the use of wormhole routing for high-speed LANs and investigated the use of

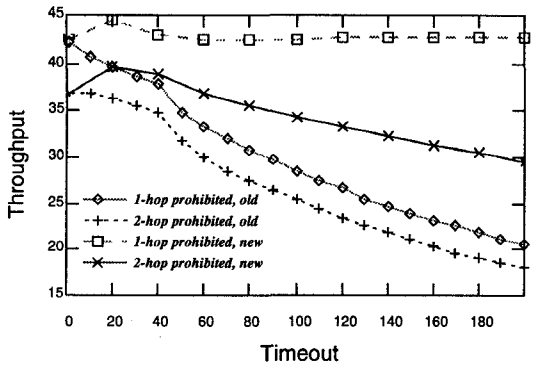

Figure 14. Throughput vs. timeout. "new" indicates the scheme in which worms deflect as soon as possible. (7×7 torus, average worm size = 50 flits.)

timeout reset. Simulations show that timeout alleviates blocking in both switches and hosts to achieve high throughput, when the network size is limited. To further improve the network performance for large scale networks, a host deflection scheme was proposed and demonstrated to be very effective. Simulation exhibits a four fold improvement in throughput, compared to that with timeout only. The host deflection scheme is easy to implement on low cost switches, which makes it quite attractive for high-speed, low-cost LANs.

Several extensions to host deflection are currently undergoing. First, we are trying to set the hop-prohibited values dynamically according to the worm size. Second, a queueing model based on the one developed in [3] is under development. The model may well give us a means to optimize the hop-prohibited parameter for any worm size. Finally, how the host buffer size affects the performance of the host deflection scheme needs to be investigated.

References

[1] J. Duato. "Improving the Efficiency of Virtual Channels with Time-Dependent Selection Functions". *Computers and Artificial Intelligence*, 13:632–650, 1994.

[2] C. J. Glass and L. M. Ni. "The Turn Model for Adaptive Routing". *Computer Architecture News*, 20:278–287, May 1992.

[3] P.-C. Hu and L. Kleinrock. "A Queueing Model for Wormhole Routing with Timeout". In *Proceedings of the 4th International Conference on Computer Communications and Networks*, pages 584–593, Las Vegas, NV, U.S., Sept. 1995.

[4] P. Kermani and L. Kleinrock. "Virtual cut-through: A New Computer Communication Switching Technique". *Computer Networks*, 3:267–289, 1979.

[5] L. Kleinrock, M. Gerla, N. Bambos, J. Cong, E. Gafni, L. Bergman, J. Bannister, S. M. T. Bujewski, P.-C. Hu, B. Kannan, B. Kwan, E. Leonardi, J. Peck, P. Palnati, and S. Walton. "The Supercomputer Supernet Testbed: A WDM Based Supercomputer Interconnect". to appear in *IEEE JSAC/JLWT* joint special issue on Multiple Wavelength Optical Technologies and Networks, 1995.

[6] L. Kleinrock, M. Gerla, N. Bambos, J. Cong, E. Gafni, L. Bergman, J. Bannister, S. Monacos, P.-C. Hu, B. Kannan, B. Kwan, J. Peck, P. Palnati, and S. Walton. "The Supercomputer Supernet (SSN): A High-speed Electro-optic Campus and Metropolitan Network". In *SPIE Optical Interconnects in Broadband Switching Architectures conference*, 1996.

[7] e. a. L. Kleinrock. "The Supercomputer Supernet: A Scalable Distributed Terabit Network". *Journal of High Speed Networks: special issue on Optical Networks*, 4(4):407–24, 1995.

[8] J. Y. Ngai and C. L. Seitz. "A Framework of Adaptive Routing in Multicomputer Networks". In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures (SPAA'89)*, pages 1–9, June 1989.

[9] L. M. Ni and P. K. McKinley. "A Survey of Wormhole Routing Techniques in Direct Networks". *Computer*, pages 62–76, Feb. 1993.

[10] D. S. Reeves and E. F. Gehringer. "Adaptive Routing for Hypercube Multiprocessors: A Performance Study". *International Journal of High Speed Computing*, pages 1–29, Mar. 1994.

[11] C. Seitz, D. Cohen, and R. Felderman. "Myrinet—A Gigabit-per-second Local-Area Network". *IEEE Micro*, 15(1):29–36, Feb. 1995.

[12] C. Seitz *et al.* "The Hypercube Communications Chip". Technical report, Dep. Computer Science, California Inst., Mar. 1985. Display File 5128:DF:85.